Arbitrary-Centered Discrete Gaussian Sampling over the Integers

Yusong Du[1,2(✉)], Baoying Fan[1], and Baodian Wei[1,2]

[1] School of Data and Computer Science, Sun Yat-sen University, Guangzhou 510006, China
duyusong@mail.sysu.edu.cn, weibd@mail.sysu.edu.cn
[2] Guangdong Key Laboratory of Information Security Technology, Guangzhou 510006, China

Abstract. Discrete Gaussian sampling over the integers, which is to sample from a discrete Gaussian distribution $\mathcal{D}_{\mathbb{Z},\sigma,\mu}$ over the integers $\mathbb{Z}$ with parameter $\sigma > 0$ and center $\mu \in \mathbb{R}$, is one of fundamental operations in lattice-based cryptography. The sampling algorithm should support a varying center μ and even a varying parameter σ, when it is used as one of the subroutines in an algorithm for sampling trapdoor lattices, or sampling from Gaussian distributions over a general n-dimensional lattice Λ. In this paper, combining the techniques in Karney's algorithm for exactly sampling the standard normal distribution, we present an exact sampling algorithm for $\mathcal{D}_{\mathbb{Z},\sigma,\mu}$ with an integer-valued parameter σ. This algorithm requires no pre-computation storage, uses no floating-point arithmetic, supports centers of arbitrary precision, and does not have any statistical discrepancy. Applying the convolution-like property of discrete Gaussian distributions, we also present an approximated sampling algorithm for $\mathcal{D}_{\mathbb{Z},\sigma,\mu}$ with a real-valued parameter σ. It also supports centers of arbitrary precision, and we show that the distribution it produces has a smaller max-log distance to the ideal distribution, as compared to Micciancio-Walter sampling algorithm, which was introduced by Micciancio et al. in Crypto 2017 for discrete Gaussian distributions with varying σ and μ over the integers.

Keywords: Lattice-based cryptography · Discrete Gaussian distribution · Rejection sampling · Exact sampling · Max-log distance

1 Introduction

Lattice-based cryptography has been accepted as a promising candidate for public key cryptography in the age of quantum computing. Discrete Gaussian sampling, which is to sample from a discrete Gaussian distribution $\mathcal{D}_{\Lambda,\sigma,\mathbf{c}}$ with

This work was supported by National Key R&D Program of China (2017YFB0802500), National Natural Science Foundations of China (Grant Nos. 61672550, 61972431), Guangdong Major Project of Basic and Applied Research (2019B030302008), Guangdong Basic and Applied Basic Research Foundation (No. 2020A1515010687) and the Fundamental Research Funds for the Central Universities (Grant No. 19lgpy217).

J. K. Liu and H. Cui (Eds.): ACISP 2020, LNCS 12248, pp. 391–407, 2020.
https://doi.org/10.1007/978-3-030-55304-3_20

parameter $\sigma > 0$ and center $\mathbf{c} \in \mathbb{R}^n$ over an n-dimensional lattice Λ, plays a fundamental role in lattice-based cryptography. Discrete Gaussian sampling is not only one of the fundamental operations in many lattice-based cryptosystems but is also at the core of security proofs of these cryptosystems [10,18,21]. It has been considered by the cryptography research community as one of the fundamental building blocks of lattice-based cryptography [19,20,22,24].

An important sub-problem of discrete Gaussian sampling, which is denoted by Sampleℤ, is to sample from a discrete Gaussian distribution $\mathcal{D}_{\mathbb{Z},\sigma,\mu}$ over the integers $\mathbb{Z}$ with parameter $\sigma > 0$ and center $\mu \in \mathbb{R}$. Since Sampleℤ is much more efficient and simpler than sampling from discrete Gaussian sampling over a general lattice, the operations involving discrete Gaussian sampling in some lattice-based cryptosystems such as [6,23,27] are nothing but Sampleℤ. A good sampling algorithm for a discrete Gaussian distribution (not necessarily over the integers $\mathbb{Z}$) should not only be efficient, but also have a negligible statistical discrepancy with the target distribution. Therefore, how to design and implement good sampling algorithms for discrete Gaussian distributions over the integers has received a lot of attentions in recent years.

The commonly used methods (techniques) for Sampleℤ are the inversion sampling (using a cumulative distribution table, CDT) [24], the Knuth-Yao method (using a discrete distribution generating (DDG) tree) [7,28], the rejection sampling [6,10,16], and the convolution technique [22,25] (based on the convolution-like properties of discrete Gaussian distributions developed by Peikert in [24]).

The first Sampleℤ algorithm, which was given by Gentry et al. in [10], uses rejection sampling. Although this algorithm supports varying parameters (including μ and σ), it is not very efficient, since it requires about 10 trials on average before outputting an integer in order to get a negligible statistical distance to the target discrete Gaussian distribution.[1]

Most of improved Sampleℤ algorithms are designed only for the case where center μ is fixed in advance, such as [6,11,14,15,28,29]. It is necessary to consider generic Sampleℤ algorithms that support varying parameters. Sampling from discrete Gaussian distributions over the integers is also usually one of the subroutines in discrete Gaussian sampling algorithms for distributions over a general n-dimensional lattice Λ. Examples include the Sample$\mathbb{D}$ algorithm [10] for distributions over an n-dimensional lattice of a basis $\mathbf{B} \in \mathbb{R}^n$, Peikert's algorithm for distributions over a q-ary integer lattice $\Lambda \subseteq \mathbb{Z}^n$ [24], and Gaussian sampling algorithms for trapdoor lattices [9,20]. A Sampleℤ algorithm should support a varying center μ, and even a varying parameter σ, if it is used in these cases.

1.1 Related Work

In 2016, Karney proposed an algorithm for sampling exactly (without statistical discrepancy) from a discrete Gaussian distribution over the integers $\mathbb{Z}$ [16].

[1] The number of trials could be decreased by using more cryptographically efficient measures, like Rényi divergence [2,26].

This algorithm uses no floating-point arithmetic and does not need any precomputation storage. It allows the parameters (including σ and μ) to be arbitrary rational numbers of finite precision. This may be the second generic Sample$\mathbb{Z}$ algorithm since the one given by Gentry et al. in [10].

In 2017, Micciancio and Walter developed a new Sample$\mathbb{Z}$ algorithm [22]. We call this algorithm Micciancio-Walter sampling algorithm. It extends and generalizes the techniques that were used in the sampler proposed by Pöeppelmann et al. [25]. Micciancio-Walter algorithm is also generic, i.e., it can be used to sample from discrete Gaussian distributions with arbitrary and varying parameters of specified (finite) precision. Moreover, Aguilar-Melchor et al. also designed a non-centered CDT algorithm with reduced size of precomputation tables [1].

More recently, it was suggested that the Bernoulli sampling, introduced by Ducas et al. in [6] for centered discrete Gaussian distributions over the integers, could be improved by using the polynomial approximation technique. Specifically, the rejection operation in the Bernoulli sampling can be performed very efficiently by using an approximated polynomial [3,32]. The polynomial (its coefficients) can be determined in advance, and it also allows us to sample from Gaussian distributions with a varying center $\mu \in [0, 1)$. Combining the convolution-like property of discrete Gaussian distributions [22,24,25], a non-centered Bernoulli sampling could be further extended to a generic sampling algorithm for any discrete Gaussian distribution over the integers. Howe et al. further presented a modular framework [12] for generating discrete Gaussians with arbitrary and varying parameters, which incorporates rejection sampling, the polynomial approximation technique, and the sampling technique used in Falcon signature [27].

Another alternative method of discrete Gaussian sampling over the integers is to sample from the (continuous) standard normal distribution, and then obtain the samples of discrete Gaussian distributions by rejection sampling [31]. This method is very efficient and supports discrete Gaussian distributions with arbitrary and varying parameters, although it relies on floating-point arithmetic (even involving logarithmic function and trigonometric function due to sampling from the standard normal distribution). In fact, a more simple and efficient method is to replace discrete Gaussians with rounded Gaussians [13], which are the nearest integers of sample values from the continuous normal distribution, but the security analysis of rounded Gaussians is only confined to the cryptosystems like Bliss signature.

Except Karney's sampling algorithm, the existing algorithms for sampling from $\mathcal{D}_{\mathbb{Z},\sigma,\mu}$ with varying parameters, either rely on floating-point arithmetic, such as the algorithms in [10,32], or require a large amount of precomputation storage, such as Micciancio-Walter sampling algorithm [22] and the one given by Aguilar-Melchor et al. [1]. The sampler presented by Barthe et al. in [3] supports a varying $\mu \in \mathbb{R}$ but not a varying parameters σ. It needs to perform another polynomial approximation procedure for the new σ.

Furthermore, except Karney's sampling algorithm, those algorithms mentioned above are all the approximated algorithms, which only produce samples

approximately from the target distribution. They usually involve complicated and careful security analysis based on statistical measures (e.g. Rényi divergence [2,26], max-log distance [22], relative error [30]), since attaining a negligible statistical measure to the ideal Gaussian distribution may be crucial for lattice-based cryptography, especially for signatures [17] and lattice trapdoors [10], to provide zero-knowledgeness. Therefore, it is interesting to consider exact sampling algorithms in lattice-based cryptography. The security analysis based on statistical measures for exact algorithms can be simplified or even be omitted.

1.2 Our Contribution

On one hand, we note that Karney's sampling algorithm [16] is exact sampling algorithm, but it allows only the case where σ and μ are rational numbers of specified (finite) precision. In this paper, for an integer-valued parameter σ, we present an exact sampling algorithm for $\mathcal{D}_{\mathbb{Z},\sigma,\mu}$. This algorithm requires no pre-computation storage, uses no floating-point arithmetic and supports a varying μ of arbitrary precision. On the other hand, although Micciancio-Walter sampling algorithm [22] supports varying parameters (including μ and σ) with specified (finite) precision, its base sampler requires a large amount of precomputation storage. Based on our proposed exact algorithm, applying the convolution-like property of discrete Gaussian distributions we give an approximated sampling algorithm for $\mathcal{D}_{\mathbb{Z},\sigma,\mu}$ with a real-valued parameter σ. It requires no pre-computation storage, and supports a varying μ of arbitrary precision. We show that the distribution it produces has a smaller max-log distance to the ideal distribution $\mathcal{D}_{\mathbb{Z},\sigma,\mu}$, as compared to the distribution produced by Micciancio-Walter sampling algorithm.

1.3 Techniques

Let σ be a positive integer, $\mu \in [0,1)$ be a real number of arbitrary precision, and x be a non-negative integer. We give an algorithm for exactly generating a Bernoulli random value which is true with probability

$$\exp\left(-t\frac{2x+t}{2x+2}\right),$$

where t is in the form of $(y - s\mu)/\sigma$, $s \in \{-1, 1\}$, and y is an integer taken uniformly from $[(1+s)/2, \sigma + (1+s)/2)$. In our exact sampling algorithm for discrete Gaussian distributions, which is based on the rejection sampling, we show that the rejection operation can be performed by repeatedly using this algorithm of generating the Bernoulli random value.

In fact, this algorithm is adapted from Karney's algorithm for exactly generating the Bernoulli random value (see Algorithm B in [16]). Karney also used it as the rejection operation in his algorithm for standard normal distribution as well as discrete Gaussian distributions over the integers. However, t is only regarded as a random deviate from $[0,1)$ in Karney's algorithm. The value of t

corresponds exactly to the fraction part of the prospective output and can be obtained directly as a random number in case of standard normal distribution, while the determination of the value of t in the case of discrete Gaussian distributions needs the computation with respect to the parameters σ and μ. In order to maintain the exactness, μ is only allowed to be a rational number for sampling from discrete Gaussian distributions, which limits the functionality of Karney's algorithm.

2 Preliminaries

2.1 Notation

We denote the set of real numbers by $\mathbb{R}$, the set of integers by $\mathbb{Z}$, and the set of non-negative integers by $\mathbb{Z}^+$. We extend any real function $f(\cdot)$ to a countable set A by defining $f(A) = \sum_{x \in A} f(x)$ if it exists. The Gaussian function on $\mathbb{R}$ with parameter $\sigma > 0$ and $\mu \in \mathbb{R}$ evaluated at $x \in \mathbb{R}$ can be defined by $\rho_{\sigma,\mu}(x) = \exp\left(-\frac{(x-\mu)^2}{2\sigma^2}\right)$. For real $\sigma > 0$ and $\mu \in \mathbb{R}$, the discrete Gaussian distribution over $\mathbb{Z}$ is defined by $\mathcal{D}_{\mathbb{Z},\sigma,\mu}(x) = \rho_{\sigma,\mu}(x)/\rho_{\sigma,\mu}(\mathbb{Z})$ for $x \in \mathbb{Z}$. Similarly, a discrete Gaussian distribution over $\mathbb{Z}^+$ is defined by $\mathcal{D}_{\mathbb{Z}^+,\sigma}(x) = \rho_{\sigma,\mu}(x)/\rho_{\sigma,\mu}(\mathbb{Z}^+)$. By convention, the subscript μ is omitted when it is taken to be 0.

2.2 Rejection Sampling

Rejection sampling is a basic method (technique) used to generate observations from a distribution [5]. It generates sampling values from a target distribution X with arbitrary probability density function $f(x)$ by using a proposal distribution Y with probability density function $g(x)$. The basic idea is that one generates a sample value from X by instead sampling from Y and accepting the sample from Y with probability

$$f(x)/Mg(x),$$

repeating the draws from Y until a value is accepted, where M is a constant such that $f(x) \leq Mg(x)$ for all values of x in the support of X. If $f(x) \leq Mg(x)$ for all x then the rejection sampling procedure produces exactly, with enough replicates, the distribution of X. In fact, $f(x)$ is allowed to be only a relative probability density function. Rejection sampling can be used to sample from a distribution X whose normalizing constant is unknown as long as the support of Y includes the support of x.

2.3 Karney's Algorithm

Karney's exact sampling algorithm for discrete Gaussian distributions, which is described as Algorithm 1, uses rejection sampling, and it is a discretization of his algorithm for sampling exactly from the normal distribution. Here, parameter σ and μ are in the set of rational numbers $\mathbb{Q}$.

Algorithm 1. [16] Sampling $\mathcal{D}_{\mathbb{Z},\sigma,\mu}$ for $\sigma, \mu \in \mathbb{Q}$ and $\sigma > 0$

Input: rational number σ and μ
Output: an integer z according to $\mathcal{D}_{\mathbb{Z},\sigma,\mu}$
1: select $k \in \mathbb{Z}^+$ with probability $\exp(-k/2) \cdot (1 - \exp(-1/2))$.
2: accept k with probability $\exp\left(-\frac{1}{2}k(k-1)\right)$, otherwise, **goto** step 1.
3: set $s \leftarrow \pm 1$ with equal probabilities.
4: set $i_0 \leftarrow \lceil k\sigma + s\mu \rceil$ and set $x_0 \leftarrow (i_0 - (k\sigma + s\mu))/\sigma$.
5: sample $j \in \mathbb{Z}$ uniformly in $\{0, 1, 2, \cdots, \lceil \sigma \rceil - 1\}$.
6: set $x \leftarrow x_0 + j/\sigma$ and **goto** step 1 if $x \geq 1$.
7: **goto** step 1 if $k = 0$ **and** $x = 0$ **and** $s < 0$.
8: accept x with probability $\exp\left(-\frac{1}{2}x(2k+x)\right)$, otherwise **goto** step 1.
9: **return** $s(i_0 + j)$

From the perspective of rejection sampling, in Algorithm 1, we can see that step 1 and step 2 together form a rejection sampling procedure, which generates $k \in \mathbb{Z}^+$ according to

$$\mathcal{D}_{\mathbb{Z}^+,1}(k) = \rho_1(k)/\rho_1(\mathbb{Z}^+),$$

which is a discrete Gaussian distribution over the set of non-negative integers $\mathbb{Z}^+$. Then, the proposal distribution for the whole algorithm can be written as

$$g(z) = g(s(\lceil k\sigma + s\mu \rceil + j)) = \rho_1(k)/(2\lceil \sigma \rceil \rho_1(\mathbb{Z}^+))$$

with $z = s(\lceil k\sigma + s\mu \rceil + j)$. The algorithm accepts z as the returned value with probability $e^{-\frac{1}{2}x(2k+x)}$, where $x = (\lceil k\sigma + s\mu \rceil - (k\sigma + s\mu) + j)/\sigma < 1$. It is not hard to see that

$$\rho_1(k) \cdot \exp(-\frac{1}{2}x(2k+x)) = \exp(-\frac{(\lceil k\sigma + s\mu \rceil + j - s\mu)^2}{2\sigma^2}) = \rho_{\sigma,\mu}(z),$$

which guarantees the correctness of Algorithm 1.

In Algorithm 1, in order to exactly sample $k \in \mathbb{Z}^+$ with (relative) probability density $\rho_1(k) = \exp(-k^2/2)$, Karney also gave an algorithm for exactly generating a Bernoulli random value which is true with probability $1/\sqrt{e}$. Specifically, one needs $(k+1)$ Bernoulli random values from $\mathcal{B}_{1/\sqrt{e}}$ to select an integer $k \geq 0$ with probability $\exp(-k/2) \cdot (1 - \exp(-1/2))$ (step 1), then continues to generate $k(k-1)$ Bernoulli random values from $\mathcal{B}_{1/\sqrt{e}}$ to accept k with probability $\exp\left(-\frac{1}{2}k(k-1)\right)$ (step 2). Karney's algorithm for exactly generating a Bernoulli random value which is true with probability $1/\sqrt{e}$ is adapted from Von Neumann's algorithm for exactly sampling from the exponential distribution e^{-x} for real $x > 0$ (see Algorithm V and Algorihtm H in [16]).

In Algorithm 1, step 8 is implemented by using a specifically designed algorithm so that it produces no any statistical discrepancy, and we will discuss this algorithm in Sect. 3.

3 Sampling from Arbitrary-Centered Discrete Gaussians

Algorithm 2 is our proposed algorithm for sampling from arbitrary-centered discrete Gaussian distributions. We give the proof of its correctness.

Algorithm 2. Sampling from $\mathcal{D}_{\mathbb{Z},\sigma,\mu}$ with an integer-valued σ and a real-valued μ of arbitrary precision

Input: positive integer σ and $\mu \in [0, 1)$
Output: an integer z according to $\mathcal{D}_{\mathbb{Z},\sigma,\mu}$
1: select $x \in \mathbb{Z}^+$ with probability $\exp(-x/2) \cdot (1 - \exp(-1/2))$.
2: accept x with probability $\exp\left(-\frac{1}{2}x(x-1)\right)$, otherwise, **goto** step 1.
3: set $s \leftarrow \pm 1$ with equal probabilities.
4: sample $y \in \mathbb{Z}$ uniformly in $\{0, 1, 2, \cdots, \sigma - 1\}$ and set $y \leftarrow y + 1$ if $s = 1$.
5: **return** $z = s(\sigma \cdot x + y)$ with probability $\exp\left(-((y - s\mu)^2 + 2\sigma x(y - s\mu))/(2\sigma^2)\right)$, otherwise **goto** step 1.

Theorem 1. *The integer $z \in \mathbb{Z}$ output by Algorithm 2 is exactly from the discrete Gaussian distribution $\mathcal{D}_{\mathbb{Z},\sigma,\mu}$ with an integer-valued σ and a real-valued μ of arbitrary precision, if the probability*

$$\exp\left(-\frac{(y - s\mu)^2 + 2\sigma x(y - s\mu)}{2\sigma^2}\right)$$

can be calculated exactly.

Proof. From the perspective of rejection sampling, following Karney's approach (step 1 and 2 in Algorithm 1), we generate an integer x exactly from $\mathcal{D}_{\mathbb{Z}^+,1}(x) = \rho_1(x)/\rho_1(\mathbb{Z}^+)$, and use $\mathcal{D}_{\mathbb{Z}^+,1}$ as the proposal distribution. For a given positive integer σ, any $z \in \mathbb{Z}$ can be uniquely written as

$$z = s\left(\sigma x + y + \frac{1+s}{2}\right),$$

where $s \in \{-1, 1\}$, and x, y are integers such that $x \geq 0$ and $y \in [0, \sigma)$. This guarantees that the support of the distribution produced by Algorithm 2 is the set of all the integers. For simplicity, we set $y \leftarrow y+1$ if $s = 1$. Then, $z = s(\sigma x+y)$ and the target distribution density function $f(z)$ for $z \in \mathbb{Z}$ can be written as

$$f(z) = f(s(\sigma x + y)) = \frac{\rho_{\sigma,\mu}(s(\sigma x + y))}{\rho_{\sigma,\mu}(\mathbb{Z})}.$$

In Algorithm 2, for a given integer x exactly from $\mathcal{D}_{\mathbb{Z}^+,1}$, we sample $s \leftarrow \pm 1$ with equal probabilities, take $z = s(\sigma x + y)$, and then accept the value of z as

the returned value with probability $\exp\left(-((y-s\mu)^2+2\sigma x(y-s\mu))/(2\sigma^2)\right)$. It is not hard to see that

$$\begin{aligned}\rho_1(x)\cdot\exp\left(-\frac{(y-s\mu)^2+2\sigma x(y-s\mu)}{2\sigma^2}\right)&=\exp\left(-\frac{(\sigma x+y-s\mu)^2}{2\sigma^2}\right)\\&=\exp\left(-\frac{(s(\sigma x+y)-\mu)^2}{2\sigma^2}\right)\\&=\rho_{\sigma,\mu}(s(\sigma x+y)),\end{aligned}$$

which is proportional to the desired (relative) probability density. This implies that the probability of Algorithm 2 going back to step 1 is equal to a constant,

$$1-(1-\exp(-1/2))\sum_{z=-\infty}^{+\infty}\rho_{\sigma,\mu}(z)=1-(1-\exp(-1/2))\,\rho_{\sigma,\mu}(\mathbb{Z}).$$

We denote by Q_∞ this constant and let $q(z)=(1-\exp(-1/2))\cdot\rho_{\sigma,\mu}(z)$. Then, the probability that Algorithm 2 outputs an integer $z\in\mathbb{Z}$ can be given by

$$q(z)+q(z)Q_\infty+\ldots+q(z)Q_\infty^i+\ldots=q(z)\cdot\sum_{i=0}^{\infty}Q_\infty^i=\frac{\rho_{\sigma,\mu}(z)}{\rho_{\sigma,\mu}(\mathbb{Z})},$$

which shows the correctness of Algorithm 2. Since all the operations, including computing the value of probability

$$\exp\left(-\frac{(y-s\mu)^2+2\sigma x(y-s\mu)}{2\sigma^2}\right),$$

can be performed without any statistical discrepancy, and thus Algorithm 2 produces exactly the discrete Gaussian distribution $\mathcal{D}_{\mathbb{Z},\sigma,\mu}$. □

The most important problem of Algorithm 2 is to compute exaclty the value of the exponential function for a real-valued μ of arbitrary precision, and get a Bernoulli random value which is true with probability of this value. Addressing this problem is based on the following observation.

$$\begin{aligned}&\exp\left(-\frac{(y-s\mu)^2+2\sigma x(y-s\mu)}{2\sigma^2}\right)\\&\quad=\exp\left(-\frac{1}{2}\left(\frac{y-s\mu}{\sigma}\right)\left(2x+\frac{y-s\mu}{\sigma}\right)\right)\\&\quad=\left(\exp\left(-\frac{1}{2}\left(\frac{y-s\mu}{\sigma}\right)\left(\frac{2x+(y-s\mu)/\sigma}{x+1}\right)\right)\right)^{x+1}\\&\quad=\left(\exp\left(-\left(\frac{y-s\mu}{\sigma}\right)\left(\frac{2x+(y-s\mu)/\sigma}{2x+2}\right)\right)\right)^{x+1}\\&\quad=\left(\exp\left(-\tilde{y}\left(\frac{2x+\tilde{y}}{2x+2}\right)\right)\right)^{x+1},\end{aligned}$$

where $\tilde{y} = (y - s\mu)/\sigma$ and $0 \leq \tilde{y} < 1$. Therefore, we can repeatedly sample from the Bernoulli distribution $\mathcal{B}_p$ a total of $x+1$ times to obtain a Bernoulli random value which is true with our desired probability, where

$$p = \exp\left(-\tilde{y}\left(\frac{2x+\tilde{y}}{2x+2}\right)\right).$$

Sampling the Bernoulli distribution $\mathcal{B}_p$ can be accomplished by using Algorithm 3, which was proposed by Karney in [16]. The function $C(m)$ with $m = 2x+2$ in Algorithm 3 is a random selector that outputs -1, 0 and 1 with probability $1/m$, $1/m$ and $1-2/m$ respectively.

Algorithm 3. [16] Generating a Bernoulli random value which is true with probability $\exp(-t(2x+t)/(2x+2))$ with integer $x \geq 0$ and real $t \in [0,1)$

Output: a Boolean value according to $\exp(-t\frac{2x+t}{2x+2})$

1: set $u \leftarrow t$, $n \leftarrow 0$.
2: sample a uniform deviate v with $v \in [0,1)$; **goto** step 6 unless $v < u$.
3: set $f \leftarrow C(2x+2)$; if $f < 0$ **goto** step 6.
4: sample a uniform deviate $w \in [0,1)$ if $f = 0$, and **goto** step 6 unless $w < t$.
5: set $u \leftarrow v$, $n \leftarrow n+1$; **goto** step 2.
6: **return true** if n is even, otherwise **return false**.

The main idea is to sample two sets of uniform deviates $v_1, v_2, \ldots$ and $w_1, w_2, \ldots$ from $[0,1)$, and to determine the maximum value $n \geq 0$ such that

$$t > v_1 > v_2 > \ldots > v_n \quad \text{and} \quad w_i < (2x+t)/(2x+2) \quad \text{for } i = 1, 2, \ldots, n.$$

If n is even, it returns **true**, and the probability is exactly equal to

$$1 - t\left(\frac{2x+t}{2x+2}\right) + \frac{t^2}{2!}\left(\frac{2x+t}{2x+2}\right)^2 - \frac{t^3}{3!}\left(\frac{2x+t}{2x+2}\right)^3 + \ldots = \exp\left(-t\frac{2x+t}{2x+2}\right).$$

This follows from the Taylor expansion of the exponential function. Then, applying this procedure at most $k+1$ times, one can obtain a Bernoulli random value which is true with probability $\exp\left(-\frac{1}{2}t(2x+t)\right)$ for given x and t. Taking

$$t = \tilde{y} = \frac{y - s\mu}{\sigma}$$

and applying Algorithm 3, we can sample from the Bernoulli distribution $\mathcal{B}_p$ with $p = \exp\left(-\tilde{y}\left((2x+\tilde{y})/(2x+2)\right)\right)$.

The remaining issue is that we need to compare $\tilde{y}$ with a randomly generated deviate in $[0,1)$, but do not use floating-point arithmetic. This can guarantee the exactness. We observe that any real $u \in [0,1)$ of arbitrary precision can be represented by

$$u = \frac{j - sr}{\sigma}$$

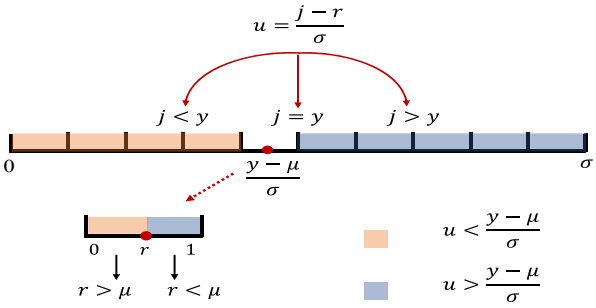

Fig. 1. Comparing $u = (j - r)/\sigma$ with $(y - \mu)/\sigma$

with given σ and s, where j is an integer from $[(1 + s)/2, \sigma + (1 + s)/2)$ and $r \in [0, 1)$ is a real number of arbitrary precision. Then, we can do the comparison as follows.

To obtain a randomly generated deviate in the form of $u = (j - sr)/\sigma \in [0, 1)$, we sample j uniformly in $\{0, 1, 2, \cdots, \sigma - 1\}$, sample a uniform deviate $r \in [0, 1)$ and set $j = j + 1$ if $s = 1$. To compare a randomly generated deviate $u = (j - sr)/\sigma$ with a given $(y - s\mu)/\sigma$, we compare j with y firstly and return the result if they are not equal. Otherwise, we compare r with μ to complete the whole procedure. Figure 1 shows the above idea in the case of $s = 1$. We summarize the whole procedure of comparison in Algorithm 4.

Algorithm 4. Compare $(y - s\mu)/\sigma$ with a randomly generated deviate $u \in [0, 1)$

Input: integers $\sigma > 0$, $s \in \{-1, 1\}$, $y \in [(1 + s)/2, \sigma + (1 + s)/2)$, and real $\mu \in [0, 1)$
Output: **true** if $u < (y - s\mu)/\sigma$ or **false** if otherwise

1: sample j uniformly in $\{0, 1, 2, \cdots, \sigma - 1\}$ and set $j \leftarrow j + 1$ if $s = 1$
2: **return** **true** if $j < y$, or **false** if $j > y$
3: sample a uniform deviate $r \in [0, 1)$
4: **return** the value of Boolean expression '$r < \mu$' if $s = -1$
5: **return** the value of Boolean expression '$r > \mu$' if $s = 1$

Note that comparing r with μ can be realized without floating-point arithmetic through bitwise operations. This implies that Algorithm 4 allows μ to be a real number of arbitrary precision and there is no case where $r = \mu$. Specifically, following the implementation of Karney's algorithm for random numbers of arbitrary precision, the comparison of two deviates is realized digit-by-digit, and each digit of a deviate is generated online according to the actual needs. To determine the relation between r and μ, we take the first digit of r and μ, respectively, denoted by r_1 and μ_1. If r_1 or μ_1 does not exist, then we generate it uniformly online. If r_1 and μ_1 are equal, then we take the second digit of r and μ, namely r_2 and μ_2, and continue to compare them. In fact, one digit could consist of only one bit or a small number of bits, such as 4 bits, 8 bits or 16 bits.

We call the number of bits in each digit the digit size, which can be specified in the source code in advance.

Finally, if each random deviate in $[0, 1)$ used in Algorithm 3 is handled in the form of $u = (j - sr)/\sigma$, then Algorithm 3 allows μ to be a real number of arbitrary precision, and we can implement it without floating-point arithmetic.

4 Applying the Convolution-Like Property

In this section, we try to extend Algorithm 2 to the case of discrete Gaussian distributions with arbitrary parameters (including σ and μ) by using the convolution-like property of discrete Gaussian distributions.

Informally, for a Gaussian distribution with a relatively large standard deviation σ, we compute two samples x_1 and x_2 with smaller variances σ_1^2 and σ_2^2. We hope that their combination $x_1 + c \cdot x_2$ with a constant c is Gaussian with variance $\sigma_1^2 + c \cdot \sigma_2^2$. Although this is not generally the case for discrete Gaussian distributions, Peikert showed that the distribution of $x_1 + c \cdot x_2$ is statistically close to discrete Gaussian distribution with variance $\sigma_1^2 + c \cdot \sigma_2^2$ under certain conditions with respect to the smoothing parameter of lattices [24]. This observation was called by Peikert the convolution-like property of discrete Gaussian distributions.

Definition 1 (a special case of [21], Definition 3.1). *Let $\epsilon > 0$ be a positive real. The smoothing parameter of lattice $\mathbb{Z}$, denoted by $\eta_\epsilon(\mathbb{Z})$, is defined to be the smallest real s such that $\rho_{1/s}(\mathbb{Z} \setminus \{0\}) \leq \epsilon$.*

Lemma 1 (Adapted from Lemma 3.3 [21]). *For any real $\epsilon > 0$, the smoothing parameter of lattice $\mathbb{Z}$ satisfies[2] $\eta_\epsilon(\mathbb{Z}) \leq \sqrt{\ln(2(1 + 1/\epsilon))/2}/\pi$.*

Here, we apply the conclusion described by Micciancio and Walter in [22] about the convolution-like property, since it deals with non-centered discrete Gaussian over the integers $\mathbb{Z}$ and uses a more cryptographically efficient measure of closeness between probability distributions, named the max-log distance.

Definition 2 [22]. *The max-log distance between two distributions $\mathcal{P}$ and $\mathcal{Q}$ over the same support S is $\Delta_{\mathrm{ML}}(\mathcal{P}, \mathcal{Q}) = \max_{x \in S} |\ln \mathcal{P}(x) - \ln \mathcal{Q}(x)|$.*

Lemma 2 (Corollary 4.2 in [22]). *Let $\sigma_1, \sigma_2 > 0$, $\sigma^2 = \sigma_1^2 + \sigma_2^2$ and $\sigma_3^{-2} = \sigma_1^{-2} + \sigma_2^{-2}$. Let $\Lambda = h \cdot \mathbb{Z}$ be a copy of the integer lattice $\mathbb{Z}$ scaled by a constant h. For any μ_1 and $\mu_2 \in \mathbb{R}$, we denote by $\tilde{\mathcal{D}}_{\mu_1+\mathbb{Z},\sigma}$ the distribution of*

$$x_1 \leftarrow x_2 + \tilde{\mathcal{D}}_{\mu_1 - x_2 + \mathbb{Z}, \sigma_1} \quad \text{with} \quad x_2 \leftarrow \tilde{\mathcal{D}}_{\mu_2+\Lambda,\sigma_2}.$$

If $\sigma_1 \geq \eta_\epsilon(\mathbb{Z})$, $\sigma_3 \geq \eta_\epsilon(\Lambda) = h \cdot \eta_\epsilon(\mathbb{Z})$, $\Delta_{\mathrm{ML}}(\mathcal{D}_{\mu_2+\Lambda,\sigma_2}, \tilde{\mathcal{D}}_{\mu_2+\Lambda,\sigma_2}) \leq \epsilon_2$ and $\Delta_{\mathrm{ML}}(\mathcal{D}_{\mu+\mathbb{Z},\sigma_1}, \tilde{\mathcal{D}}_{\mu+\mathbb{Z},\sigma_1}) \leq \epsilon_1$ for any $\mu \in \mathbb{R}$, then

$$\Delta_{\mathrm{ML}}(\mathcal{D}_{\mu_1+\mathbb{Z},\sigma}, \tilde{\mathcal{D}}_{\mu_1+\mathbb{Z},\sigma}) \leq 4\epsilon + \epsilon_1 + \epsilon_2,$$

where ϵ_1 and ϵ_2 are positive real numbers.

[2] It allows to decrease the smoothing condition by a factor of $\sqrt{2\pi}$ since the Gaussian function is defined to be $\exp(-\frac{(x-\mu)^2}{2\sigma^2})$ but not $\exp(-\pi\frac{(x-\mu)^2}{\sigma^2})$.

Definition 3 (Randomized rounding operator $\lfloor\cdot\rceil_\lambda$ [22]). *For $\mu \in [0,1)$ and a positive integer λ, the randomized rounding operator $\lfloor\mu\rceil_\lambda$ is defined by*

$$\lfloor\mu\rceil_\lambda = \lfloor 2^\lambda \mu \rfloor / 2^\lambda + B_\alpha / 2^\lambda$$

with a Bernoulli random variable B_α of parameter $\alpha = 2^\lambda \mu - \lfloor 2^\lambda \mu \rfloor$. In particular, if $\mu > 1$ then

$$\mu' \leftarrow \lfloor\mu\rfloor + \lfloor\{\mu\}\rceil_\lambda,$$

where $\lfloor\mu\rfloor$ and $\{\mu\}$ is the integer part and the fractional part of μ respectively.

Lemma 3 (Adapted from Lemma 5.3 in [22]). *Let $\lambda > \log_2 4\pi$ be a positive integer and $b = 2^\lambda$. If $\sigma \geq \eta_\epsilon(\mathbb{Z})$, then*

$$\Delta_{\mathrm{ML}}(\mathcal{D}_{\mathbb{Z},\sigma,\mu}, \tilde{\mathcal{D}}_{\mathbb{Z},\sigma,\lfloor\mu\rceil_\lambda}) \leq (\pi/b)^2 + 2\epsilon,$$

where $\lfloor\cdot\rceil_\lambda$ is the randomized rounding operator as defined above.

Combining our proposed exact algorithm (Algorithm 2), and applying the convolution-like property, namely Lemma 2, we give Algorithm 5.

Algorithm 5. Sampling $\mathcal{D}_{\mathbb{Z},\sigma,\mu}$ with $\sigma > \eta_\epsilon(\mathbb{Z})$ and $\mu \in [0,1)$

Input: $\sigma > 1$ and $\mu \in [0,1)$
Output: an integer z
1: sample $x \in \mathbb{Z}$ from $\mathcal{D}_{\mathbb{Z},2\eta_\epsilon(\mathbb{Z})}$
2: set $h = \sqrt{\sigma^2 - \lfloor\sigma\rfloor^2}/(2\eta_\epsilon(\mathbb{Z}))$
3: set $\mu' \leftarrow \lfloor\mu + hx\rceil_\lambda$
4: sample z from $\mathcal{D}_{\mathbb{Z},\lfloor\sigma\rfloor,\mu'}$ and **return** z

Theorem 2 gives the correctness of Algorithm 5 and estimates the (theoretical) max-log distance between the distribution $\tilde{\mathcal{D}}_{\mathbb{Z},\sigma,\mu}$ produced by Algorithm 5 and the ideal distribution $\mathcal{D}_{\mathbb{Z},\sigma,\mu}$. The equation that

$$\mu + \mathcal{D}_{-\mu+\mathbb{Z},\sigma} = \mathcal{D}_{\mathbb{Z},\sigma,\mu}$$

for any $\sigma > 0$ and $\mu \in \mathbb{R}$ will be repeatedly used in the proof.

Theorem 2. *Let $\lambda > \log_2 4\pi$ be a positive integer and $b = 2^\lambda$. Denote by $\tilde{\mathcal{D}}_{\mathbb{Z},\sigma,\mu}$ the probability distribution of integers that are output by Algorithm 5. If $\lfloor\sigma\rfloor > \eta_\epsilon(\mathbb{Z})$ and $\eta_\epsilon(\mathbb{Z})$ is taken to be a rational number, then we have*

$$\Delta_{\mathrm{ML}}(\mathcal{D}_{\mathbb{Z},\sigma,\mu}, \tilde{\mathcal{D}}_{\mathbb{Z},\sigma,\mu}) \leq (\pi/b)^2 + 6\epsilon,$$

by using Algorithm 1 in step 1, and using Algorithm 2 in step 4.

Proof. Let $\sigma_1 = \lfloor \sigma \rfloor$ and $\sigma_2 = 2h\eta_\epsilon(\mathbb{Z})$. Then, we have $\sigma^2 = \sigma_1^2 + \sigma_2^2$ and

$$\sigma_3 = \left(\sigma_1^{-2} + (2h\eta_\epsilon(\mathbb{Z}))^{-2}\right)^{-1/2} = \frac{\sigma_1}{\sigma} \cdot \sqrt{\sigma^2 - \sigma_1^2} \geq \sqrt{\sigma^2 - \sigma_1^2} \cdot \frac{\eta_\epsilon(\mathbb{Z})}{2\eta_\epsilon(\mathbb{Z})} = \eta_\epsilon(h\mathbb{Z}).$$

This is due to the fact that $\sigma_1/\sigma = \lfloor \sigma \rfloor / \sigma \geq 1/2$ for $\sigma > 1$. With the notation of Lemma 2, taking $\mu_1 = -\mu$, $\mu_2 = 0$, $\Lambda = h\mathbb{Z}$, $x_2 = hx$ and $x_1 \leftarrow x_2 + \tilde{\mathcal{D}}_{\mu_1 - x_2 + \mathbb{Z}, \sigma_1}$, we have $x_2 = hx \leftarrow \tilde{\mathcal{D}}_{\mu_2 + \Lambda, \sigma_2} = \tilde{\mathcal{D}}_{h\mathbb{Z}, h(2\eta_\epsilon(\mathbb{Z}))} = h \cdot \tilde{\mathcal{D}}_{\mathbb{Z}, 2\eta_\epsilon(\mathbb{Z})}$ and

$$\mu + x_1 \leftarrow \mu + (x_2 + \tilde{\mathcal{D}}_{\mu_1 - x_2 + \mathbb{Z}, \sigma_1}) = \mu + hx + \tilde{\mathcal{D}}_{-\mu - hx + \mathbb{Z}, \lfloor \sigma \rfloor} = \tilde{\mathcal{D}}_{\mathbb{Z}, \lfloor \sigma \rfloor, \mu + hx}.$$

Since $\eta_\epsilon(\mathbb{Z})$ is a rational number, in Algorithm 5, $\tilde{\mathcal{D}}_{\mathbb{Z}, 2\eta_\epsilon(\mathbb{Z})}$ can be exactly sampled (without any statistical discrepancy) via Algorithm 1, which implies that

$$\Delta_{\mathrm{ML}}(\mathcal{D}_{\mathbb{Z}, 2\eta_\epsilon(\mathbb{Z})}, \tilde{\mathcal{D}}_{\mathbb{Z}, 2\eta_\epsilon(\mathbb{Z})}) = 0.$$

In contrast, $\tilde{\mathcal{D}}_{\mathbb{Z}, \lfloor \sigma \rfloor, \mu + hx}$ can be only realized by sampling from $\mathcal{D}_{\mathbb{Z}, \lfloor \sigma \rfloor, \lfloor \mu + hx \rceil_\lambda}$ with max-log distance $\Delta_{\mathrm{ML}} \leq (\pi/b)^2 + 2\epsilon$. Applying Lemma 2, we obtain that

$$\begin{aligned} \mu + x_1 \leftarrow \mu + (x_2 + \tilde{\mathcal{D}}_{\mu_1 - x_2 + \mathbb{Z}, \sigma_1}) &= \mu + (hx + \tilde{\mathcal{D}}_{-\mu - hx + \mathbb{Z}, \lfloor \sigma \rfloor}) \\ &\approx \mu + \mathcal{D}_{-\mu + \mathbb{Z}, \sigma} = \mathcal{D}_{\mathbb{Z}, \sigma, \mu}, \end{aligned}$$

where '$\approx$' means that $\Delta_{\mathrm{ML}}(hx + \mathcal{D}_{-\mu - hx + \mathbb{Z}, \lfloor \sigma \rfloor}, \mathcal{D}_{-\mu + \mathbb{Z}, \sigma})$ is not more than

$$\begin{aligned} &4\epsilon + \Delta_{\mathrm{ML}}(\mathcal{D}_{-\mu - hx + \mathbb{Z}, \lfloor \sigma \rfloor}, \tilde{\mathcal{D}}_{-\mu - hx + \mathbb{Z}, \lfloor \sigma \rfloor}) + \Delta_{\mathrm{ML}}(\mathcal{D}_{h\mathbb{Z}, h(2\eta_\epsilon(\mathbb{Z}))}, \tilde{\mathcal{D}}_{h\mathbb{Z}, h(2\eta_\epsilon(\mathbb{Z}))}) \\ &\quad = 4\epsilon + \Delta_{\mathrm{ML}}(\mathcal{D}_{\mathbb{Z}, 2\eta_\epsilon(\mathbb{Z})}, \tilde{\mathcal{D}}_{\mathbb{Z}, 2\eta_\epsilon(\mathbb{Z})}) \\ &\qquad + \Delta_{\mathrm{ML}}(\mu + hx + \mathcal{D}_{-\mu - hx + \mathbb{Z}, \lfloor \sigma \rfloor}, \mu + hx + \tilde{\mathcal{D}}_{-\mu - hx + \mathbb{Z}, \lfloor \sigma \rfloor}) \\ &\quad \leq 4\epsilon + \Delta_{\mathrm{ML}}(\mathcal{D}_{\mathbb{Z}, 2\eta_\epsilon(\mathbb{Z})}, \tilde{\mathcal{D}}_{\mathbb{Z}, 2\eta_\epsilon(\mathbb{Z})}) + \Delta_{\mathrm{ML}}(\mathcal{D}_{\mathbb{Z}, \lfloor \sigma \rfloor, \mu + hx}, \tilde{\mathcal{D}}_{\mathbb{Z}, \lfloor \sigma \rfloor, \mu + hx}) \\ &\quad \leq 4\epsilon + ((\pi/b)^2 + 2\epsilon). \\ &\quad = 6\epsilon + (\pi/b)^2. \end{aligned}$$

This completes the proof. □

The distribution produced by Algorithm 5 has a smaller max-log distance to the ideal distribution, as compared to one produced by Micciancio-Walter sampling algorithm, since both step 1 and step 4 can be implemented exactly, and do not lead to any statistical discrepancy. In contrast, the two corresponding steps in Micciancio-Walter algorithm are approximated ones (see Sect. 5 in [22]). The statistical discrepancy they produce must be counted in the total statistical discrepancy.

For instance, we take $\epsilon = 2^{-112}$ and $\eta_\epsilon(\mathbb{Z}) = 2$, as $\sqrt{\ln(2(1 + 1/\epsilon))/2}/\pi \leq 2$ when $\epsilon = 2^{-112}$. We take $\lambda = 30$ and $b = 2^{30}$, which results in $(\pi/b)^2 \leq 2^{-56}$. Then, we have $\Delta_{\mathrm{ML}}(\mathcal{D}_{\mathbb{Z}, \sigma, \mu}, \tilde{\mathcal{D}}_{\mathbb{Z}, \sigma, \mu}) \leq 2^{-56} + 6 \cdot 2^{-112}$. The practical max-log distance should also include the statistical discrepancy due to the floating-point operations in step 2. One can just follow the argument at the end of Sect. 5.3 in [22].

Moreover, the first step in Micciancio-Walter algorithm requires sampling from a centered discrete Gaussian distribution $\mathcal{D}_{\mathbb{Z},\sigma_{\max}}$ with a possible varying parameter $\sigma_{\max}$, which should be determined before sampling according to the desired distribution. In Algorithm 5, however, step 1 is just to sample from a centered discrete Gaussian distribution with a fixed parameter $2\eta_\epsilon(\mathbb{Z}) = 4$. This means that Algorithm 5 has a simpler form than Micciancio-Walter algorithm.

5 Experimental Results

Karney's sampling algorithm for discrete Gaussian distributions over the integers $\mathbb{Z}$ can be realized by using C++ library 'RandomLib'[3], in which the source code of his algorithm is encapsulated as a .hpp file named "DiscreteNormal.hpp". "RandomLib" also supports the generation and some basic operations of random numbers of arbitrary precision.

On a laptop computer (Intel i7-6820 hq, 8 GB RAM), using the g++ compiler and enabling -O3 optimization option, we tested our Algorithm 2. The source code was based on the adaptation of 'DiscreteNormal.hpp' as well as the runtime environment provided by 'RandomLib'. For discrete Gaussian distribution $\mathcal{D}_{\mathbb{Z},\sigma,\mu}$ with σ from 4 to 2^{20} and μ uniformly from $[0, 1)$ of precision 128 bits, combining Algorithms 3 and 4, one could get about 5.0×10^6 samples per second by using Algorithm 2. It has almost the same performance as Karney's algorithm.

We also tested the performance of the Micciancio-Walter algorithm with the same parameters. Using this algorithm, one could get about 1.3×10^6 integers per second. We implemented its base sampler with the CDT-based method, which required an amount of extra memory, and took $\lambda = 8$ and $b = 16$. This guarantees the max-log distance to the ideal distribution is not more than 2^{-52}.

We note that Micciancio-Walter algorithm has a constant execution time for given parameters if its base sampler is a constant-time algorithm. However, our Algorithm 2 as well as Algorithm 5 is not a constant-time one, and it seems to be inherently costly to turn into a constant-time one due to the fact that Algorithm 2 is always probabilistically rejecting samples. Therefore, an open question is how to make Algorithm 2 constant-time and be protected against side-channel attacks [4,8].

6 Conclusion

For an integer-valued parameter σ, there exists an exact sampling algorithm for $\mathcal{D}_{\mathbb{Z},\sigma,\mu}$. It requires no precomputation storage, uses no floating-point arithmetic, supports a varying μ of arbitrary precision, and does not have any statistical discrepancy. Applying the convolution-like property of discrete Gaussian distributions, it can be further extended to an approximated sampling algorithm for $\mathcal{D}_{\mathbb{Z},\sigma,\mu}$ with a real-valued parameter σ. The extended algorithm also supports centers of arbitrary precision and it produces a distribution with a smaller max-log distance to the ideal distribution, as compared to Micciancio-Walter sampling algorithm.

[3] 'RandomLib' is available at http://randomlib.sourceforge.net/.

References

1. Aguilar-Melchor, C., Albrecht, M.R., Ricosset, T.: Sampling from arbitrary centered discrete Gaussians for lattice-based cryptography. In: Gollmann, D., Miyaji, A., Kikuchi, H. (eds.) ACNS 2017. LNCS, vol. 10355, pp. 3–19. Springer, Cham (2017). https://doi.org/10.1007/978-3-319-61204-1_1
2. Bai, S., Lepoint, T., Roux-Langlois, A., Sakzad, A., Stehlé, D., Steinfeld, R.: Improved security proofs in lattice-based cryptography: using the rényi divergence rather than the statistical distance. J. Cryptol. **31**(2), 610–640 (2018)
3. Barthe, G., Belaïd, S., Espitau, T., Fouque, P., Rossi, M., Tibouchi, M.: GALACTICS: Gaussian sampling for lattice-based constant- time implementation of cryptographic signatures, revisited. In: Cavallaro, L., Kinder, J., Wang, X., Katz, J. (eds.) Proceedings of the 2019 ACM SIGSAC CCS, pp. 2147–2164. ACM, New York (2019)
4. Groot Bruinderink, L., Hülsing, A., Lange, T., Yarom, Y.: Flush, gauss, and reload – a cache attack on the BLISS lattice-based signature scheme. In: Gierlichs, B., Poschmann, A.Y. (eds.) CHES 2016. LNCS, vol. 9813, pp. 323–345. Springer, Heidelberg (2016). https://doi.org/10.1007/978-3-662-53140-2_16
5. Devroye, L.: Non-Uniform Random Variate Generation. Springer, New York (1986)
6. Ducas, L., Durmus, A., Lepoint, T., Lyubashevsky, V.: Lattice signatures and bimodal Gaussians. In: Canetti, R., Garay, J.A. (eds.) CRYPTO 2013. LNCS, vol. 8042, pp. 40–56. Springer, Heidelberg (2013). https://doi.org/10.1007/978-3-642-40041-4_3
7. Dwarakanath, N.C., Galbraith, S.D.: Sampling from discrete Gaussians for lattice-based cryptography on a constrained device. Appl. Algebra Eng. Commun. Comput **25**(3), 159–180 (2014)
8. Espitau, T., Fouque, P.A., Gérard, B., Tibouchi, M.: Side-channel attacks on bliss lattice-based signatures: exploiting branch tracing against strongswan and electromagnetic emanations in microcontrollers. In: Thuraisingham, B., Evans, D., Malkin, T., Xu, D. (eds.) Proceedings of the 2017 ACM SIGSAC CCS, pp. 1857–1874. ACM, New York (2017)
9. Genise, N., Micciancio, D.: Faster Gaussian sampling for trapdoor lattices with arbitrary modulus. In: Nielsen, J.B., Rijmen, V. (eds.) EUROCRYPT 2018. LNCS, vol. 10820, pp. 174–203. Springer, Cham (2018). https://doi.org/10.1007/978-3-319-78381-9_7
10. Gentry, C., Peikert, C., Vaikuntanathan, V.: Trapdoors for hard lattices and new cryptographic constructions. In: Ladner, R., Dwork, C. (eds.) Proceedings of the 2008 ACM SIGACT STOC, pp. 197–206. ACM, New York (2008)
11. Howe, J., Khalid, A., Rafferty, C., Regazzoni, F., O'Neill, M.: On practical discrete Gaussian samplers for lattice-based cryptography. IEEE Trans. Comput. **67**(3), 322–334 (2018)
12. Howe, J., Prest, T., Ricosset, T., Rossi, M.: Isochronous Gaussian sampling: from inception to implementation. In: Ding, J., Tillich, J.-P. (eds.) PQCrypto 2020. LNCS, vol. 12100, pp. 53–71. Springer, Cham (2020). https://doi.org/10.1007/978-3-030-44223-1_4
13. Hülsing, A., Lange, T., Smeets, K.: Rounded Gaussians. In: Abdalla, M., Dahab, R. (eds.) PKC 2018. LNCS, vol. 10770, pp. 728–757. Springer, Cham (2018). https://doi.org/10.1007/978-3-319-76581-5_25
14. Karmakar, A., Roy, S.S., Reparaz, O., Vercauteren, F., Verbauwhede, I.: Constant-time discrete Gaussian sampling. IEEE Trans. Comput. **67**(11), 1561–1571 (2018)

15. Karmakar, A., Roy, S.S., Vercauteren, F., Verbauwhede, I.: Pushing the speed limit of constant-time discrete Gaussian sampling. A case study on the falcon signature scheme. In: Aitken, R. (ed.) Proceedings of the 56th DAC, pp. 1–6. ACM, New York (2019)
16. Karney, C.F.: Sampling exactly from the normal distribution. ACM Trans. Math. Softw. **42**(1), 3:1–3:14 (2016)
17. Lyubashevsky, V.: Lattice signatures without trapdoors. In: Pointcheval, D., Johansson, T. (eds.) EUROCRYPT 2012. LNCS, vol. 7237, pp. 738–755. Springer, Heidelberg (2012). https://doi.org/10.1007/978-3-642-29011-4_43
18. Lyubashevsky, V., Peikert, C., Regev, O.: On ideal lattices and learning with errors over rings. J. ACM **60**(6), 43:1–43:35 (2013)
19. Lyubashevsky, V., Prest, T.: Quadratic time, linear space algorithms for Gram-Schmidt orthogonalization and gaussian sampling in structured lattices. In: Oswald, E., Fischlin, M. (eds.) EUROCRYPT 2015. LNCS, vol. 9056, pp. 789–815. Springer, Heidelberg (2015). https://doi.org/10.1007/978-3-662-46800-5_30
20. Micciancio, D., Peikert, C.: Trapdoors for lattices: simpler, tighter, faster, smaller. In: Pointcheval, D., Johansson, T. (eds.) EUROCRYPT 2012. LNCS, vol. 7237, pp. 700–718. Springer, Heidelberg (2012). https://doi.org/10.1007/978-3-642-29011-4_41
21. Micciancio, D., Regev, O.: Worst-case to average-case reductions based on Gaussian measures. SIAM J. Comput. **37**(1), 267–302 (2007)
22. Micciancio, D., Walter, M.: Gaussian sampling over the integers: efficient, generic, constant-time. In: Katz, J., Shacham, H. (eds.) CRYPTO 2017. LNCS, vol. 10402, pp. 455–485. Springer, Cham (2017). https://doi.org/10.1007/978-3-319-63715-0_16
23. Naehrig, M., et al.: Frodokem learning with errors key encapsulation. https://frodokem.org/files/FrodoKEM-specification-20171130.pdf. Accessed 28 Feb 2020
24. Peikert, C.: An efficient and parallel Gaussian sampler for lattices. In: Rabin, T. (ed.) CRYPTO 2010. LNCS, vol. 6223, pp. 80–97. Springer, Heidelberg (2010). https://doi.org/10.1007/978-3-642-14623-7_5
25. Pöppelmann, T., Ducas, L., Güneysu, T.: Enhanced lattice-based signatures on reconfigurable hardware. In: Batina, L., Robshaw, M. (eds.) CHES 2014. LNCS, vol. 8731, pp. 353–370. Springer, Heidelberg (2014). https://doi.org/10.1007/978-3-662-44709-3_20
26. Prest, T.: Sharper bounds in lattice-based cryptography using the Rényi divergence. In: Takagi, T., Peyrin, T. (eds.) ASIACRYPT 2017. LNCS, vol. 10624, pp. 347–374. Springer, Cham (2017). https://doi.org/10.1007/978-3-319-70694-8_13
27. Prest, T., et al.: Falcon: fast-fourier lattice-based compact signatures over NTRU. https://falcon-sign.info/. Accessed 20 Feb 2020
28. Sinha Roy, S., Vercauteren, F., Verbauwhede, I.: High precision discrete Gaussian sampling on FPGAs. In: Lange, T., Lauter, K., Lisoněk, P. (eds.) SAC 2013. LNCS, vol. 8282, pp. 383–401. Springer, Heidelberg (2014). https://doi.org/10.1007/978-3-662-43414-7_19
29. Saarinen, M.J.O.: Arithmetic coding and blinding countermeasures for lattice signatures. J. Cryptogr. Eng. **8**, 71–84 (2018)
30. Walter, M.: Sampling the integers with low relative error. In: Buchmann, J., Nitaj, A., Rachidi, T. (eds.) AFRICACRYPT 2019. LNCS, vol. 11627, pp. 157–180. Springer, Cham (2019). https://doi.org/10.1007/978-3-030-23696-0_9

31. Zhao, R.K., Steinfeld, R., Sakzad, A.: COSAC: COmpact and Scalable Arbitrary-Centered discrete Gaussian sampling over integers. In: Ding, J., Tillich, J.-P. (eds.) PQCrypto 2020. LNCS, vol. 12100, pp. 284–303. Springer, Cham (2020). https://doi.org/10.1007/978-3-030-44223-1_16
32. Zhao, R.K., Steinfeld, R., Sakzad, A.: FACCT: fast, compact, and constant-time discrete Gaussian sampler over integers. IEEE Trans. Comput. **69**(1), 126–137 (2020)